Setting Expectations for Performance Portability between Companion Accelerator and Manycore Systems

John M Levesque
Director
Cray's Supercomputing Center of Excellence
CTO Office

Outline

- **Look at recent research in Europe**
  - Tuning the implementation of the radiation scheme ACRANEB2
    - Jacob W. Poulsen and Per Berg *IT department, DMI*
    - *Most of the slides I will be using were generated by Jacob and Per*
- **Conclusions**

Real World Climate Model Issue

- **ACRANEB2 is the radiation scheme used in the IFS code**
  - Currently the radiation scheme is used intermittently, researchers would like to use it every time step
  - Highly compute intensive unlike other parts of climate modeling which tend to be memory bandwidth limited
- **Radiation scheme ported to KNL and Nvidia**
  - Then code was optimized for KNL using OpenMP (SPMD)
  - Then code was optimized further for P100 using OpenACC
- **Performance tests performed on KNL, P100 and Broadwell**

Intermezzo: Musings on performance

- Think of a programming language and a parallel programming model as a short-hand notation for generating specific code for a given target.
- Do not buy the appealing idea that you can construct efficient programs solely by using the abstractions of programming languages and parallel programming models.
- Sorry to say but you have to understand how the target architecture works if performance truly matters to you and you might instead think of the process of writing code as a process where you try to hint the compiler etc. in the right direction towards a given target architecture.
- ...and again if performance matters to you:
- Think of a programming language and a parallel programming model as a short-hand notation for generating specific code for a given target.

Intermezzo: Musings on performance

- Abstractions are very appealing from a computer scientific point of view but don't get fooled. Abstractions are never free, 2D arrays can be too much of an abstraction if performance is key.

```
!-  2D abstractions too complicated -------

    1 do k=2,kmax
    2   k1 = k+off1
    3   k2 = k+off2
    4   t(1:nc,k) = t(1:nc,k) + A(k)*(B(1:nc,k1)-B(1:nc,k2))
    5 enddo
```

- With dynamic nc the compiler vectorizes nc-loop:

  (4): (col. 7) remark: LOOP WAS VECTORIZED
- With static nc, the compiler vectorizes the k-loop:

  (1): (col 7) remark: LOOP WAS VECTORIZED
- Alas, assembler inspection revealed that gather operations were generated and runtine experiences confirm this.
- ...

Intermezzo: Musings on performance

- Programming languages and parallel programming models have several options when it comes to architectural features

  (Brent Leback et al, cug2013):
  - Hide
  - Virtualize
  - Expose

- OpenMP has a good reputation productivitywise – why ?
- OpenMP has a bad reputation performancewise – why ?
- OpenMP is a highly abstract model so very easy to use and misuse

- MPI has a bad reputation productivitywise – why ?
- MPI has a good reputation performancewise – why ?
- MPI exposes the separate nodes, the distributed memory and all "network" transfers explicitly so the programmer will have to consider how to deal with these details while implementing the program

Refactoring of legacy code

1. Establish a solid **reference** (test case and source code) that reproduces the necessary results.
2. Establish **build** and **run env.** to ease repetition and reproducibility.
3. Ensure proper **threading**, i.e. SPMD approach
   - requires transition to Fortran90 assumed-shape and trimming stack memory usage
   - contiguous data
4. Strive towards a **minimal implementation**, including:
   - Reduce memory overhead
   - Reduce stack pressure: local tmp 2D/3D vars into 1D/2D vars or even scalars
   - Largest stack arrays moved to the heap; proper NUMA initialization of heap arrays
   - Collapsing loops over the outermost index
   - Symbolic algebraic reduction using pen&paper
   - Assuring no side effects in local functions (**`pure`** in Fortran)
   - Declare constants as constants (**`parameter`** in Fortran), not as variables
   - Push all branching out of the loops
5. Continued refactoring is to shuffle computations around to **maximize parallel exposure** (playing with data structures and loops)
   - Identify computational patterns with (e.g. reduction and prefix-sum) and without (SIMD-suitable loops) dependencies
   - Re-organize heavy loops to constant trip-count
   - ...

Tuning for short latency capacity per compute unit

Some P100 specs: 56 SM (streaming multiprocessor) per GPU

32 DP cuda cores per SM

Scheduling unit is a **warp** [device limit: max 64 warps per SM]

Always 32 **threads** per warp [i.e. max 2048 active threads per SM]

Thread block [device limit: max 32 blocks per SM]:

Block size is #thread [device limit: max 1024 threads per block]

Default block has **128** threads and 4 warps

Blocks are “tunable” wrt #warps and #threads per block

[device limits: max 1024 threads and max 32 warps per block]

Registers: 256KB register file per SM

[device limits: max 65536 32bit regs per SM,

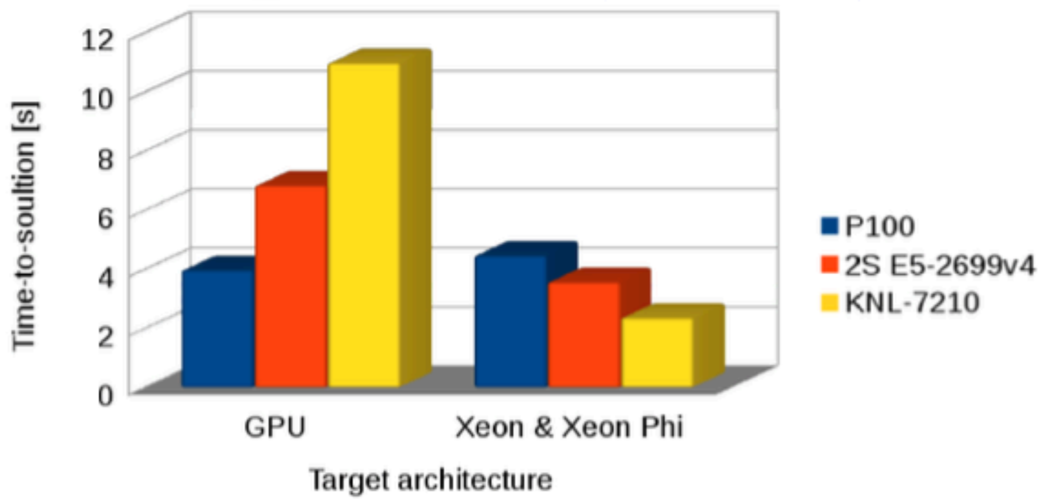
max 65536 registers per block,

max 255 registers per thread]

So, tuning for short latency capacity on P100 is about keeping #registers per thread low

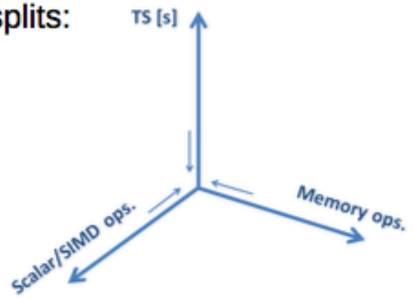
Measure is **occupancy:** #active threads in percent of device limit

Portable vs competitive performance



Note 1: **X** and **G codes** are essentially the same, except for the splits:

One must be able to hide the memory latencies resulting from extra memory transfers required to bind the smaller parts.



Note 2: **X** and **G codes** still has prefix-sum loops:

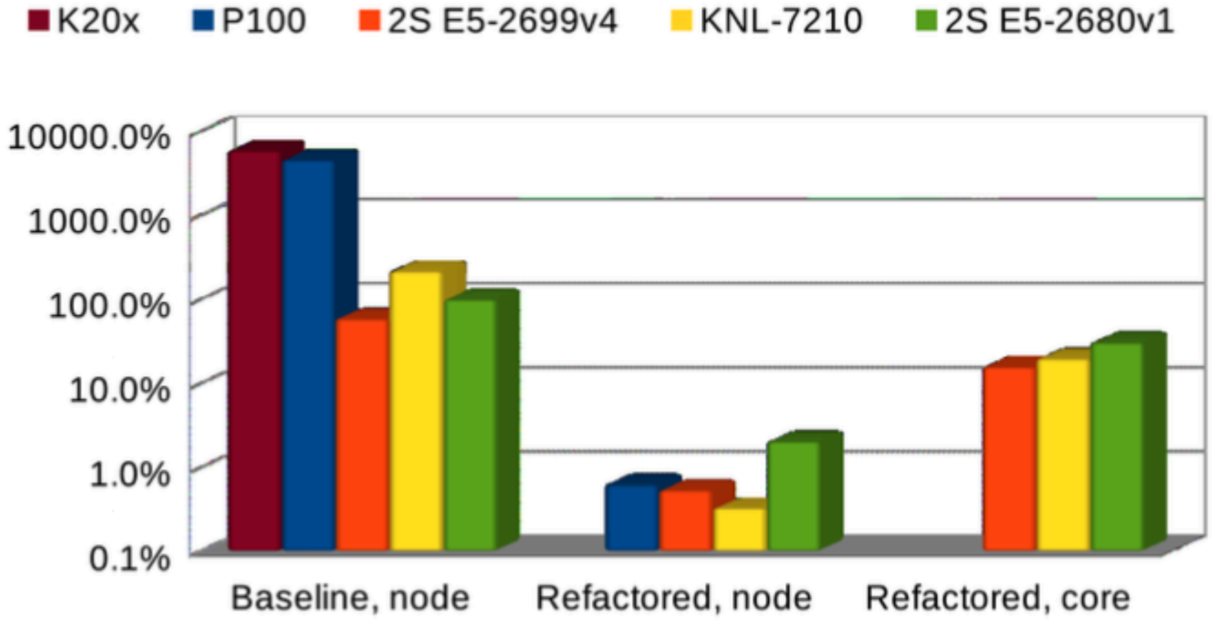
High scalar/SIMD might hurt performance.

Effect of refactoring

Refactoring is increasingly more important the newer the hardware
– legacy codes might even run slower on more modern HW

Refactoring is even more important for the high throughput architectures

Timings of TRANST in full ACRANEB2 relative to baseline code on SNB:



Hardware Counters for both X and G on Broadwell

TRANST3 Running on Broadwell	KNL Version - X	13.7%peak(DP)	GPU Version - G	3.1%peak(DP)
HW FP Ops Usertime	2,704.935M /sec	20,260,160,826	497.003M/sec	13,489,461,656
Total SP Ops	13.356M/ sec	100,035,301	6.621M/sec	179,715,564
Total-DP-ops	2,691.579M /sec	20,160,125,525	490.381M/sec	13,309,746,092
Computational-intensity	1.1 ops/ cycle	3.81 ops/ref	0.25 ops/cycle	1.52 ops/ref
MFLOPS(aggregate)	2,704.94M/ sec		497.00M/sec	
TLButilization	3088.61 refs/miss	6.03 avg uses	540.74 refs/miss	1.06 avg uses
D1cachehit,missratios	93.6% hits	6.4% misses	95.5% hits	4.5% misses
D1cacheutilization	15.71refs/ miss	1.96 avg hits	22.38refs/miss	2.80 avg hits
D2cachehit,missratio	59.6% hits	40.4% misses	79.8% hits	20.2% misses
D1+D2cachehit,miss	97.4% hits	2.6% misses	99.1% hits	.1% misses
D1+D2 cacheutilization	38.94 refs/ miss	4.87 avg hits	110.78 refs/miss	13.85 avg hits
D2toD1bandwidth	1,546.057MiB/sec	12,142,593,984 bytes	440.315MiB/sec	12,531,398,336 bytes

Hardware Counters for both X and G on Broadwell

TRANST3 Running on Broadwell	KNL Version - X	13.7%peak(DP)	GPU Version - G	3.1%peak(DP)
HW FP Ops Usertime	2,704.935M/sec	20,260,160,826	497.003M/sec	13,489,461,656
Total SP Ops	13.356M/sec	100,035,301	6.621M/sec	179,715,564
Total-DP-ops	2,691.579M/sec	20,160,125,525	490.381M/sec	13,309,746,092
Computational-intensity	1.1 ops/cycle	3.81 ops/ref	0.25 ops/cycle	1.52 ops/ref
MFLOPS(aggregate)	2,704.94M/sec		497.00M/sec	
TLButilization	3088.61 refs/miss	6.03 avg uses	540.74 refs/ miss	1.06 avg uses
D1cachehit,missratios	93.6% hits	6.4% misses	95.5% hits	4.5% misses
D1cacheutilization	15.71refs/miss	1.96 avg hits	22.38refs/miss	2.80 avg hits
D2cachehit,missratio	59.6% hits	40.4% misses	79.8% hits	20.2% misses
D1+D2cachehit,miss	97.4% hits	2.6% misses	99.1% hits	.1% misses
D1+D2 cacheutilization	38.94 refs/miss	4.87 avg hits	110.78 refs/ miss	13.85 avg hits
D2toD1bandwidth	1,546.057MiB/sec	12,142,593,984 bytes	440.315MiB/sec	12,531,398,336 bytes

Results of X and G code running on Broadwell – 8.2 Seconds with memory analysis

X

```
  Samp% | Samp | Imb. |  Imb. | MEM_LOAD_UOPS_RETIRED | RESOURCE_STALLS | Group
        |      | Samp | Samp% |    :HIT_LFB:precise=2 |            :ALL |  Function=[MAX10]
        |      |      |       |                       |                 |   Source
        |      |      |       |                       |                 |    Line

 100.0% | 94.0 |   -- |    -- |            37,600,846 |  18,471,464,610 | Total
|-----------------------------------------------------------------------------
|  70.2% | 66.0 |   -- |    -- |            26,400,588 |  13,003,862,681 | USER
||----------------------------------------------------------------------------
||  70.2% | 66.0 |   -- |    -- |            26,400,588 |  13,003,862,681 | acraneb_transt3$acraneb3__clone_31615_1501851751_2_
3|        |      |      |       |                       |                 |  acraneb2.f90
4|        |      |      |       |                       |                 |   line.130
```

G

```
  Samp% | Samp | Imb. |  Imb. | MEM_LOAD_UOPS_RETIRED | RESOURCE_STALLS | Group
        |      | Samp | Samp% |    :HIT_LFB:precise=2 |            :ALL |  Function=[MAX10]
        |      |      |       |                       |                 |   Source
        |      |      |       |                       |                 |    Line

 100.0% | 44.0 |   -- |    -- |            17,600,396 |  40,138,321,643 | Total
|-----------------------------------------------------------------------------
|  70.5% | 31.0 |   -- |    -- |            12,400,279 |  30,022,721,717 | USER
||----------------------------------------------------------------------------
||  70.5% | 31.0 |   -- |    -- |            12,400,279 |  30,022,721,717 | acraneb_transt3$acraneb3_
3|        |      |      |       |                       |                 |  acraneb3.F90
||||==========================================================================
```

Results of X and G code running on KNL with memory analysis

```
  Samp% | Samp | Imb. |  Imb. | NO_ALLOC_CYCLES | MEM_UOPS_RETIRED | Group
        |      | Samp | Samp% |            :ALL |   :L2_MISS_LOADS |  Function=[MAX10]
        |      |      |       |                 |       :precise=2 |   Source
        |      |      |       |                 |                  |    Line
|  50.0% |  2.0 |    -- |    -- |   4,050,957,765 |           80,025 | USER
||-----------------------------------------------------------------------------
||  50.0% |  2.0 |    -- |    -- |   4,050,957,765 |           80,025 | acraneb_transt3$acraneb3__clone_3707_1501852873_
3|        |      |      |       |                 |                  |  acraneb2.f90
4|        |      |      |       |                 |                  |   line.130
|==============================================================================
```

X

```
  Samp% | Samp | Imb. |  Imb. | NO_ALLOC_CYCLES | MEM_UOPS_RETIRED | Group
        |      | Samp | Samp% |            :ALL |   :L2_MISS_LOADS |  Function=[MAX10]
        |      |      |       |                 |       :precise=2 |   Source
        |      |      |       |                 |                  |    Line

 100.0% | 22.0 |    -- |    -- |  95,250,152,816 |          880,198 | Total
|-----------------------------------------------------------------------------
|  90.9% | 20.0 |    -- |    -- |  90,890,511,782 |          800,181 | USER
||----------------------------------------------------------------------------
||  86.4% | 19.0 |    -- |    -- |  90,799,881,468 |          760,155 | acraneb_transt3$acraneb3_
3|        |      |      |       |                 |                  |  acraneb3.F90
||||--------------------------------------------------------------------------
```

G

Code Optimized for KNL

```
 100.0% | 1,271.608331 |          -- | Total
|------------------------------------------------------------------------------
| 100.0% | 1,271.602597 |         1.0 | foo_
| 100.0% | 1,271.602595 |         1.0 |  main_
3  99.9% | 1,270.601680 |         2.0 |   radia_dwarf$radia_m_
4  99.9% | 1,270.463249 |         1.0 |    acraneb2$acraneb2_m_
5  99.9% | 1,270.463240 |         2.0 |     acraneb_transt3$acraneb3_
6  99.9% | 1,270.463238 |          -- |      acraneb_transt3$acraneb3_.LOOP.1.li.238 (160,000)
7  99.9% | 1,270.176018 |          -- |       acraneb_transt3$acraneb3_.LOOP.3.li.305 (40)
8  99.2% | 1,260.805241 |          -- |        acraneb_transt3$acraneb3_.LOOP.6.li.485 (81)
||||||||---------------------------------------------------------------------
9||||||||  25.4% |   323.354494 | 518,400,000.0 | zcdel0$acraneb3_
9||||||||  22.7% |   288.227424 | 518,400,000.0 | ztdel1$acraneb3_
9||||||||  15.0% |   190.522204 | 518,400,000.0 | zcdelta1$acraneb3_
9||||||||  14.4% |   182.970033 | 518,400,000.0 | ztdelta1$acraneb3_
9||||||||  11.1% |   141.669216 | 518,400,000.0 | zcdelta2$acraneb3_
9||||||||  10.5% |   134.061872 | 518,400,000.0 | ztdelta2$acraneb3_
|===============================================================================
```

Code Optimized for GPU

```
Table 1:  Function Calltree View

  Time% |        Time |        Calls | Calltree

 100.0% | 1,177.513999 |           -- | Total
|-----------------------------------------------------------------------------
| 100.0% | 1,177.508949 |          1.0 | foo_
| 100.0% | 1,177.508947 |          1.0 |  main_
3  99.8% | 1,174.984080 |          2.0 |   radia_dwarf$radia_m_
4  99.8% | 1,174.845679 |          1.0 |    acraneb2$acraneb2_m_
5  99.8% | 1,174.845669 |          2.0 |     acraneb_transt3$acraneb3_
6  99.8% | 1,174.845665 |           -- |      acraneb_transt3$acraneb3_.LOOP.0001.li.210 (2)
||||||-----------------------------------------------------------------------
7||||||  27.1% |   319.289605 |           -- | acraneb_transt3$acraneb3_.LOOP.0014.li.469 (80,000)
8||||||  27.1% |   319.157763 |           -- |  acraneb_transt3$acraneb3_.LOOP.0016.li.502 (40)
9||||||  26.4% |   311.138577 |           -- |   acraneb_transt3$acraneb3_.LOOP.0019.li.567 (81)
10|||||  26.4% |   311.138577 | 518,400,000.0 |    zcdel0$acraneb3_
7||||||  24.8% |   291.704217 |           -- | acraneb_transt3$acraneb3_.LOOP.0026.li.657 (80,000)
8||||||  24.8% |   291.496028 |           -- |  acraneb_transt3$acraneb3_.LOOP.0028.li.701 (40)
9||||||  24.0% |   283.149544 |           -- |   acraneb_transt3$acraneb3_.LOOP.0031.li.803 (81)
||||||||||-------------------------------------------------------------------
10||||||||   13.5% |   159.425912 | 518,400,000.0 | ztdelta1$acraneb3_
10||||||||   10.5% |   123.723632 | 518,400,000.0 | ztdelta2$acraneb3_
||||||||||===================================================================
7||||||  24.0% |   282.573841 |           -- | acraneb_transt3$acraneb3_.LOOP.0032.li.832 (80,000)
8||||||  24.0% |   282.452776 |           -- |  acraneb_transt3$acraneb3_.LOOP.0034.li.865 (40)
9||||||  23.4% |   274.949679 |           -- |   acraneb_transt3$acraneb3_.LOOP.0037.li.934 (81)
10|||||  23.4% |   274.949679 | 518,400,000.0 |    ztdel1$acraneb3_
7||||||  13.5% |   158.979422 |           -- | acraneb_transt3$acraneb3_.LOOP.0002.li.218 (80,000)
8||||||  13.5% |   158.788738 |           -- |  acraneb_transt3$acraneb3_.LOOP.0004.li.254 (40)
9||||||  12.9% |   151.482292 |           -- |   acraneb_transt3$acraneb3_.LOOP.0007.li.323 (81)
10|||||  12.9% |   151.482292 | 518,400,000.0 |    zcdelta1$acraneb3_
7||||||  10.4% |   122.140792 |           -- | acraneb_transt3$acraneb3_.LOOP.0008.li.347 (80,000)
8||||||  10.4% |   121.957475 |           -- |  acraneb_transt3$acraneb3_.LOOP.0010.li.380 (40)
9||||||   9.8% |   114.865167 |           -- |   acraneb_transt3$acraneb3_.LOOP.0013.li.445 (81)
10|||||   9.8% |   114.865167 | 518,400,000.0 |    zcdelta2$acraneb3_
|============================================================================
```

Further tuning – 2

Sketch of latest GPU target code: nproma & 12-way split, **GN code**

- C1.1
- C1.2
- C2.1
- C2.2
- C3
- Crest
- T1.1
- T1.2
- T2.1
- T2.2
- T3
- Trest

C3 chunk as an example:

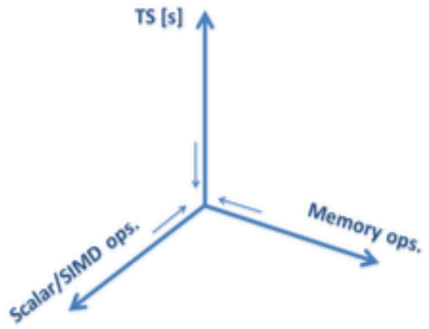
```
!$acc parallel
Jlon-loop, stride nproma:
    Jlev-loop:
        i=1,nproma:
            Preparation: 8 (5 RO, 3 WO)
    Jlev1-loop:
        Jlev2-loop:
            i=1,nproma:
                Prefix-sum: 12 (3+2+1 RO, 6 WO)
        Jlev2-loop:
            i=1,nproma:
                Fat loop: 8+1 (6+1 RO, 1 WR)
```

Further tuning – 2

GN code: scalar/SIMD ratio decreases, so does occupancy: **25 - 43.75%**

Result is that T2S decreases by 1.7x: T2S = **2.32 s**

Part	regs/thread	theoretical occupancy [%]
C1.1	96	31.25
C1.2	72	43.75
C2.1	94	31.25
C2.2	72	43.75
C3	118	25
Crest	94	31.25
T1.1	96	31.25
T1.2	72	43.75
T2.1	94	31.25
T2.2	72	43.75
T3	112	25
Trest	80	37.50



Loop Table for NPROMA =32 run

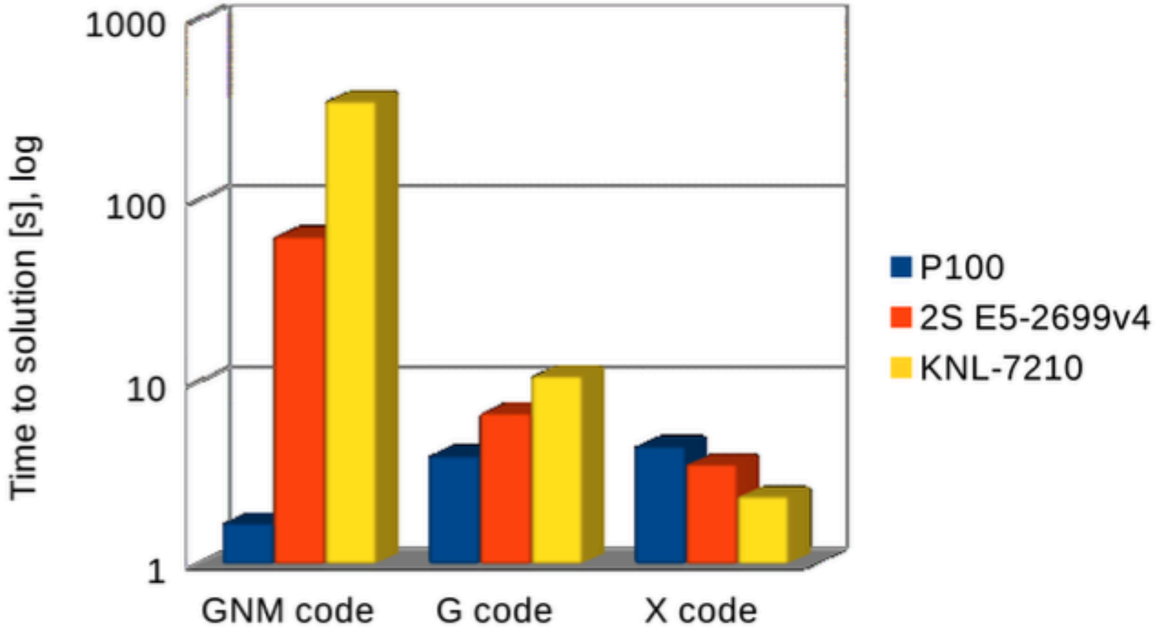
```
Table 2:  Inclusive and Exclusive Time in Loops (from -hprofile_generate)

   Loop |    Loop Incl |      Time |  Loop Hit |    Loop |  Loop |  Loop | Function=/.LOOP[.]
   Incl |         Time |     (Loop |           |   Trips | Trips | Trips |
  Time% |              |     Adj.) |           |     Avg |   Min |   Max |
|-----------------------------------------------------------------------------
| 131.8% | 2,090.202939 |  0.000008 |          1 |     2.0 |     2 |     2 | acraneb_transt3$acraneb3_.LOOP.0001.li.179
|  23.8% |   378.109848 |  0.003810 |          2 |   625.0 |   625 |   625 | acraneb_transt3$acraneb3_.LOOP.0050.li.607
|  23.8% |   378.047825 |  0.302535 |      1,250 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0053.li.633
|  23.7% |   375.755567 |  0.038828 |     50,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0060.li.713
|  23.7% |   375.716739 | 44.147064 |  4,050,000 |   128.0 |   128 |   128 | acraneb_transt3$acraneb3_.LOOP.0061.li.719
|  23.4% |   370.373284 |  0.003828 |          2 |   625.0 |   625 |   625 | acraneb_transt3$acraneb3_.LOOP.0121.li.1223
|  23.4% |   370.312884 |  0.308230 |      1,250 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0124.li.1249
|  23.2% |   368.024567 |  0.039692 |     50,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0131.li.1330
|  23.2% |   367.984875 | 44.315884 |  4,050,000 |   128.0 |   128 |   128 | acraneb_transt3$acraneb3_.LOOP.0132.li.1336
|  15.2% |   240.320211 |  0.014168 |          2 | 2,500.0 | 2,500 | 2,500 | acraneb_transt3$acraneb3_.LOOP.0073.li.807
|  15.1% |   240.165708 |  1.060907 |      5,000 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0076.li.832
|  15.0% |   238.259467 |  0.014077 |          2 | 2,500.0 | 2,500 | 2,500 | acraneb_transt3$acraneb3_.LOOP.0002.li.188
|  15.0% |   238.104327 |  1.013176 |      5,000 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0005.li.213
|  14.9% |   235.604348 |  0.131438 |    200,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0083.li.896
|  14.8% |   235.472911 | 48.302278 | 16,200,000 |    32.0 |    32 |    32 | acraneb_transt3$acraneb3_.LOOP.0084.li.902
|  14.7% |   233.629111 |  0.134209 |    200,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0012.li.278
|  14.7% |   233.494902 | 46.806381 | 16,200,000 |    32.0 |    32 |    32 | acraneb_transt3$acraneb3_.LOOP.0013.li.283
|  13.0% |   206.404953 |  0.013350 |          2 | 2,500.0 | 2,500 | 2,500 | acraneb_transt3$acraneb3_.LOOP.0097.li.1024
|  13.0% |   206.301616 |  1.019964 |      5,000 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0100.li.1047
|  12.8% |   202.361857 |  0.145688 |    200,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0107.li.1102
|  12.8% |   202.216170 | 50.037572 | 16,200,000 |    32.0 |    32 |    32 | acraneb_transt3$acraneb3_.LOOP.0108.li.1107
|  11.5% |   182.050951 |  0.013106 |          2 | 2,500.0 | 2,500 | 2,500 | acraneb_transt3$acraneb3_.LOOP.0026.li.402
|  11.5% |   181.948361 |  1.003706 |      5,000 |    40.0 |    40 |    40 | acraneb_transt3$acraneb3_.LOOP.0029.li.424
|  11.2% |   178.034804 |  0.131957 |    200,000 |    81.0 |    81 |    81 | acraneb_transt3$acraneb3_.LOOP.0036.li.481
|  11.2% |   177.902847 | 48.839249 | 16,200,000 |    32.0 |    32 |    32 | acraneb_transt3$acraneb3_.LOOP.0037.li.486
```

Tuning for short latency capacity per compute unit

How is the **GN code** performing on the Xeons ?

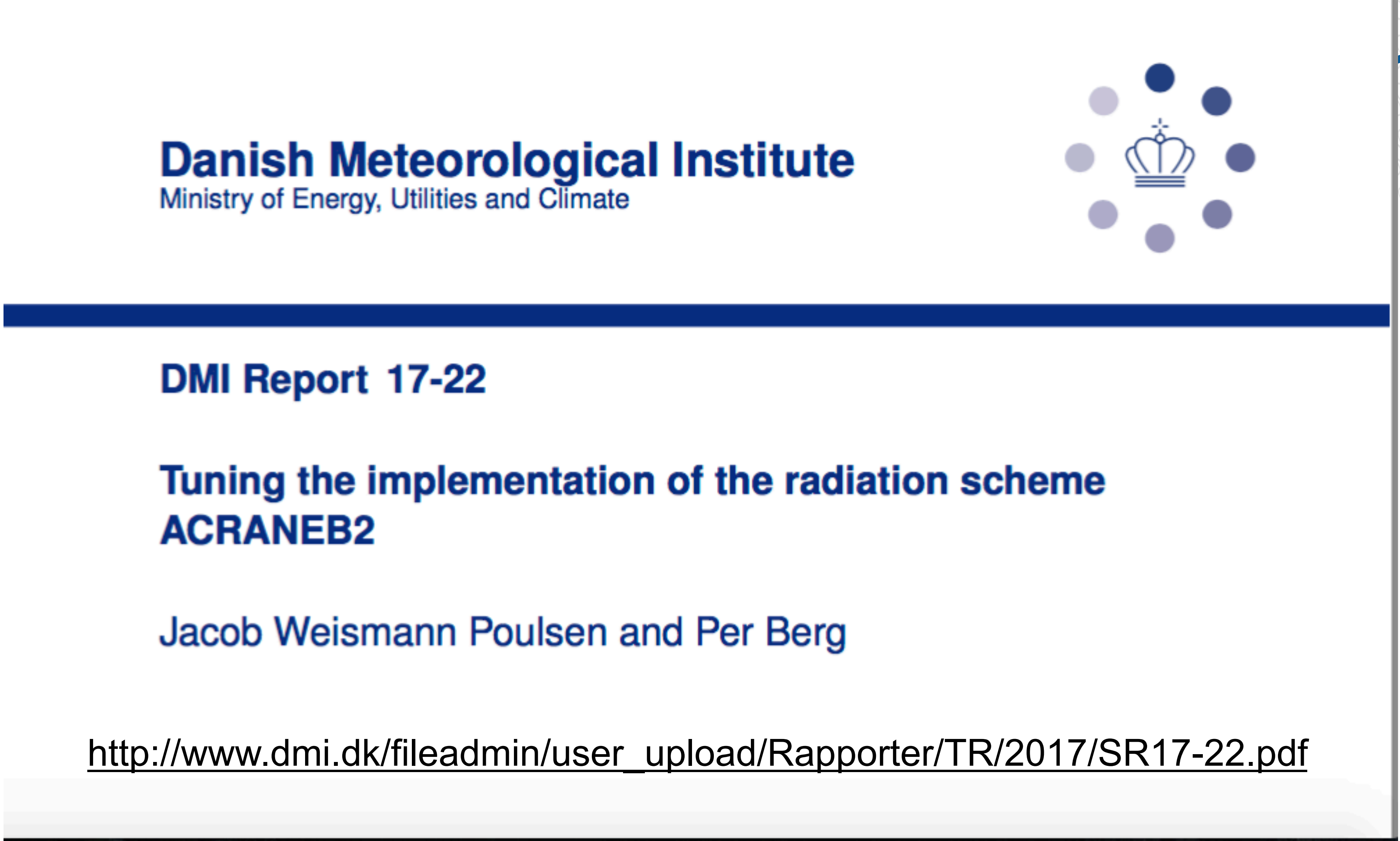
… Actually, performance drops **A LOT!!!**

- T2S more than **1 min** on BDW and **5 min** on KNL vs **2.3 secs** on P100



Conclusions

- The author's of the aforementioned paper and I believe that performance is more important than productivity
- For this radiation scheme
  - The best performance on the GPU does not perform well on KNL and state-of-the-art Xeon
  - The best performance on KNL performs well on Xeon and okay on the GPU.
- Register spilling to Memory on the GPU hurts performance and rewriting to minimize spills generates code that significantly abuses cache on the Xeon and KNL systems
- Perftools is indicating that the principal difference in the two versions of the code is the cache utilization
- Memory analysis tool shows that the L2 cache misses on KNL for the G code is ten times what it is for the X code
- Memory analysis tool shows that the stalls attributed to memory access are twice as big for the G code on Broadwell than the X code



Danish Meteorological Institute

Ministry of Energy, Utilities and Climate

DMI Report 17-22

Tuning the implementation of the radiation scheme ACRANEB2

Jacob Weismann Poulsen and Per Berg

http://www.dmi.dk/fileadmin/user_upload/Rapporter/TR/2017/SR17-22.pdf